Как мы создавали Data Management Platform в Ozon

Евгений Чмель





Agenda

1. Что такое DMP
2. Интерфейс конструктора сегментов
3. Архитектура

1

Что такое DMP

DMP — платформа для построения сегментов пользователей для таргетинга.

Сегмент — набор пользователей (user_id и session_id), который может быть сформирован по различным правилам (фильтрам).

Некоторые из них:

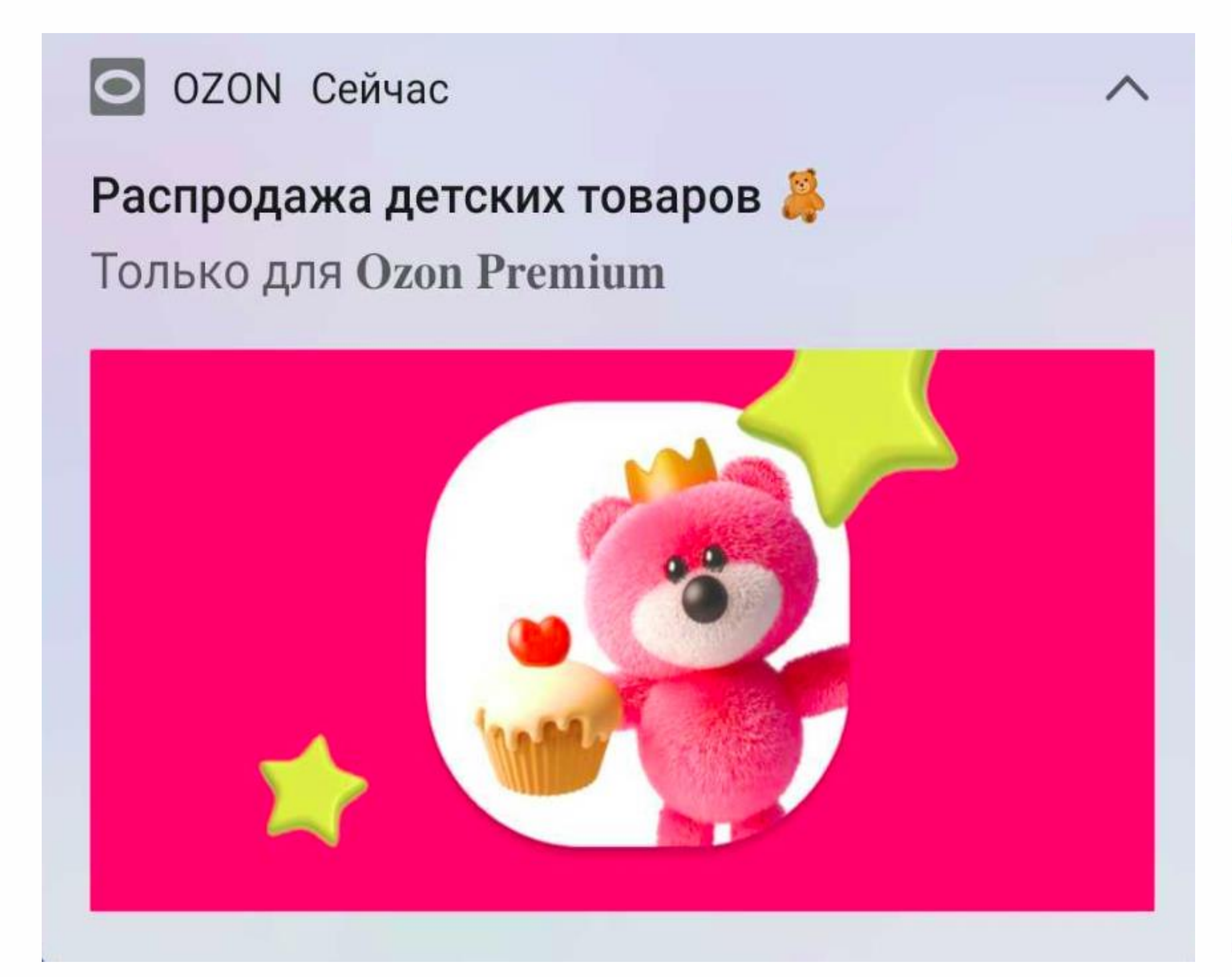
- Город
- Бренд
- Категория
- Поисковый запрос
- Тип платформы

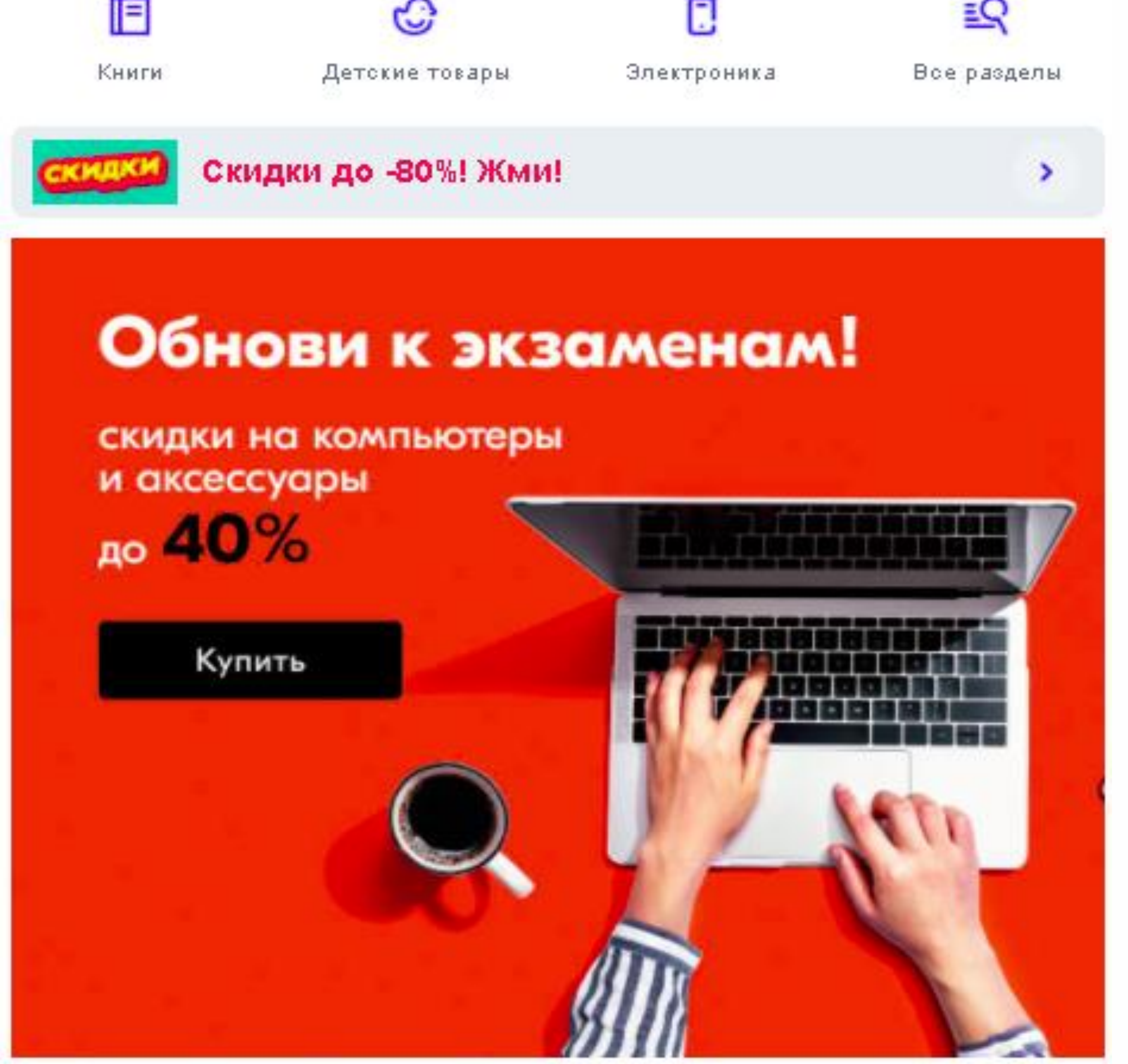
Атрибуты сегмента:

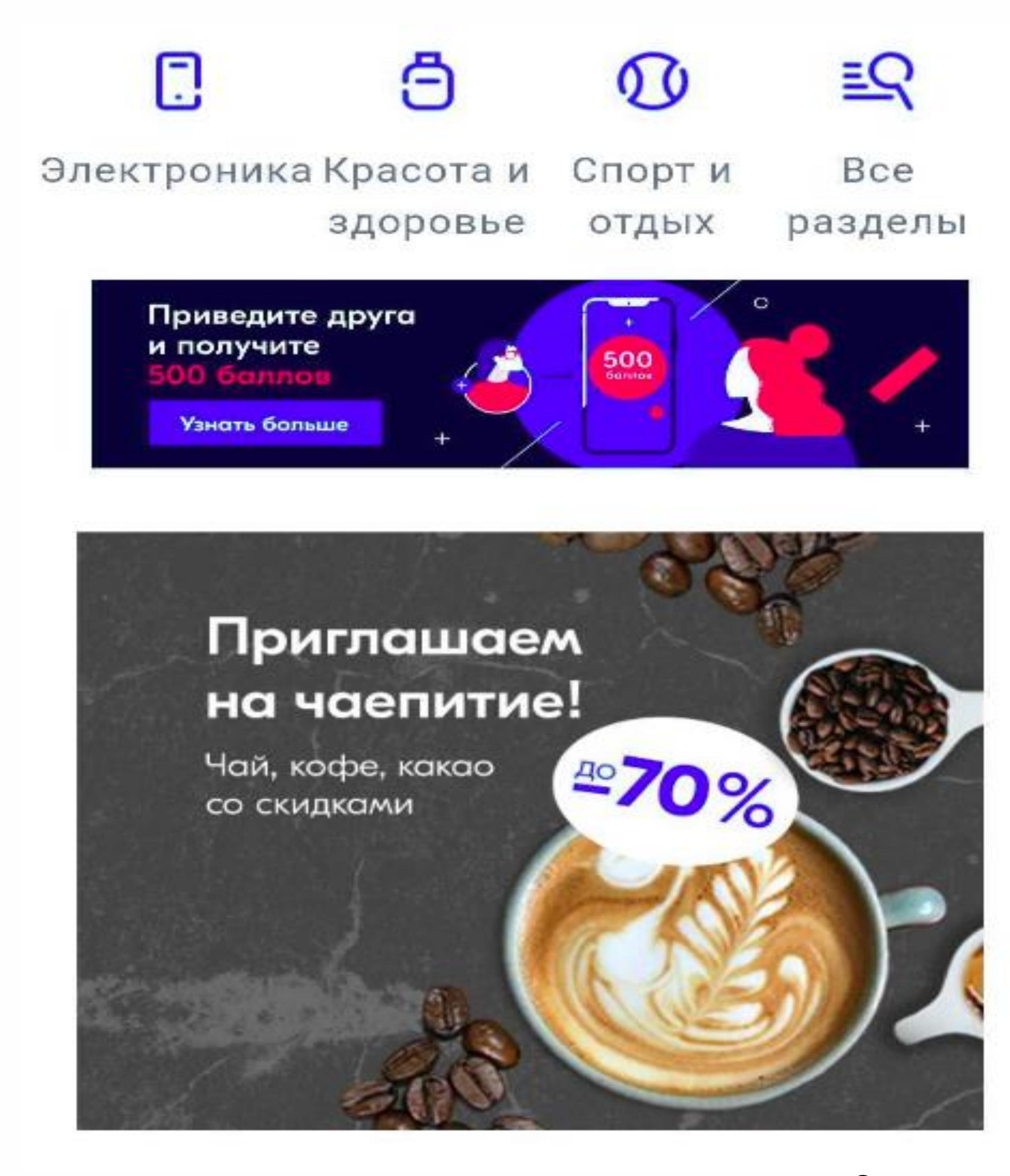
- ID – номер сегмента
- Name – название
- TTL – время экспирации
- Type – динамический или статический
- Date_interval – интервал существования сегмента
- Users_quantity – количество пользователей в сегменте

Примеры использования сегментов

- Отправляем нотификации, рассылаем письма
- Показываем рекомендации, баннеры, страницы с товарами
- Ценообразование через маркетинговые акции







Изначально сегменты создавались вручную:

- Поступала заявка на новый сегмент
- Разработчик реализовывал механику

Основной недостаток: тратилось много времени на написание кода. Количество заявок увеличивалось.

Решили делать конструктор сегментов.

На каких данных делать автоматизацию сборки сегментов?

Сегменты можно строить на ивентах, которые создаются во время пользовательской активности.


Эти ивенты есть в двух хранилищах: **ClickHouse и HDFS**

В итоге остановились на **ClickHouse.**

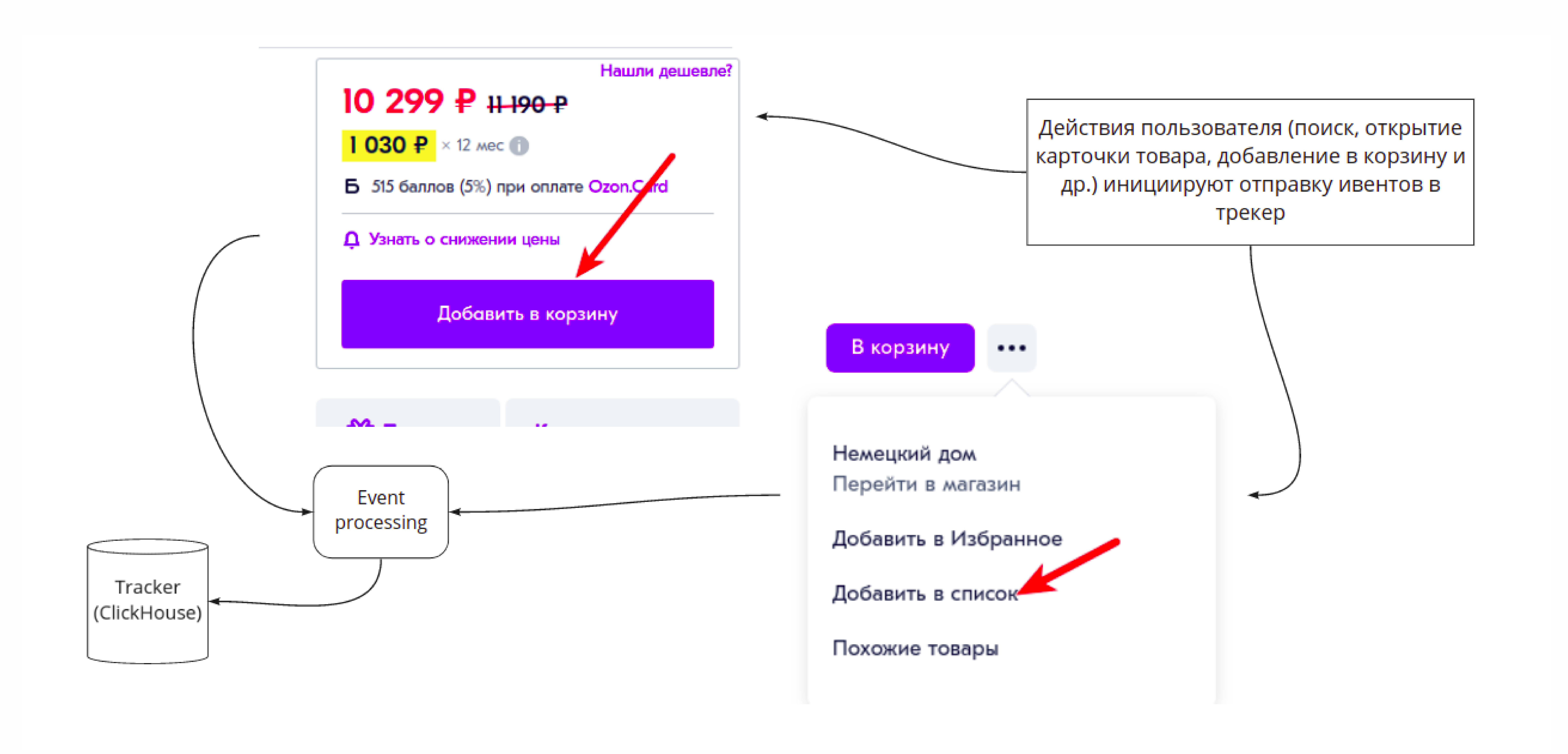
Пробовали делать запросы через Spark поверх HDFS, но скорость выполнения была в более чем 30 раз ниже, чем у ClickHouse.

Позже добавили еще один источник (**Vertica**) для построения сегментов по заказам.

Каждые сутки делается выгрузка по заказам из сервисной БД OMS (OrderManagementSystem) в Vertica.

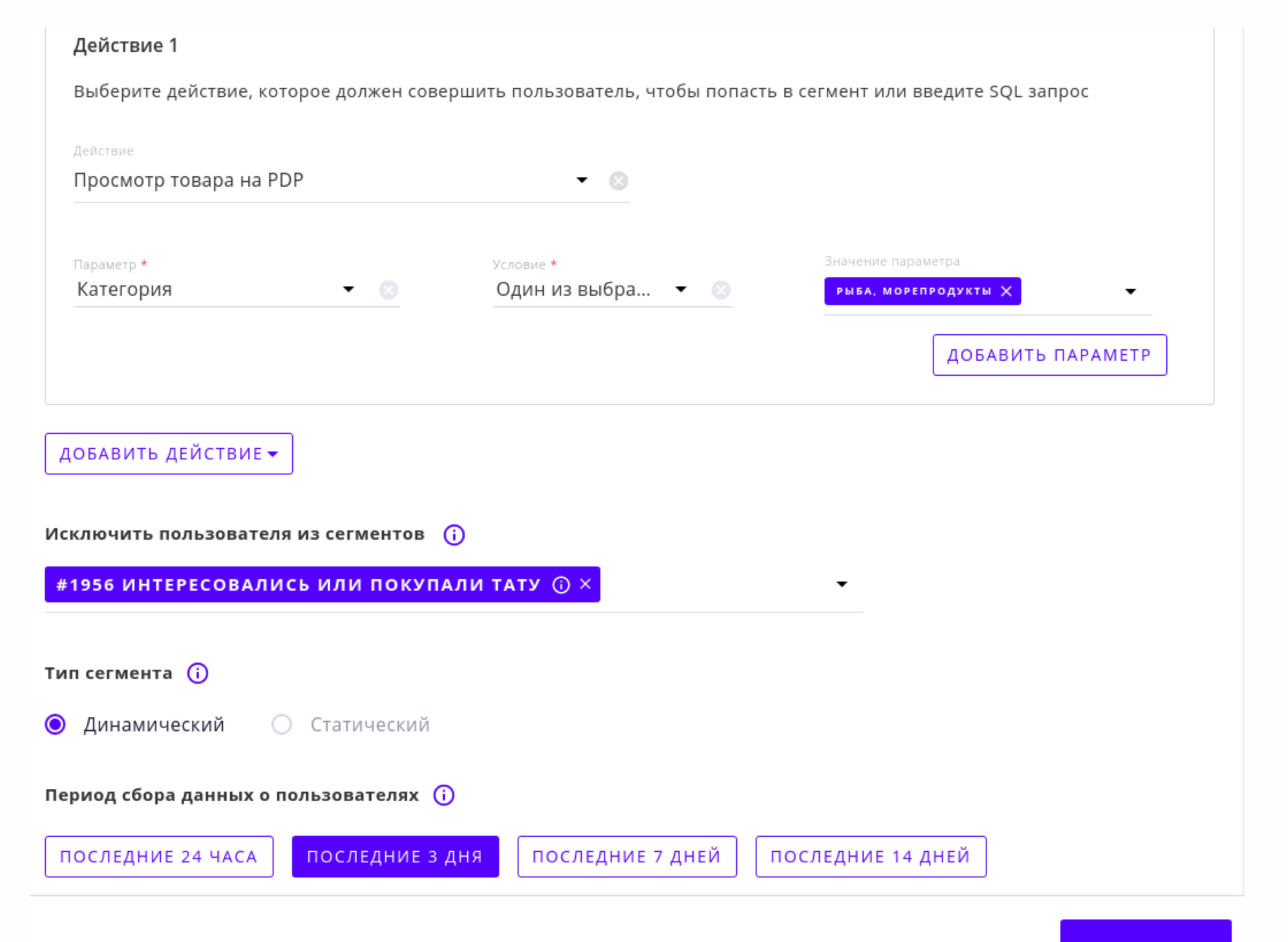


Событие — клик на товаре, просмотр, добавление в корзину и др.


Нашли дешевле?
10 299 ₽ 11 190 ₽
1 030 ₽ × 12 мес
Б 515 баллов (5%) при оплате Ozon.Card
Узнать о снижении цены
Добавить в корзину
Действия пользователя (поиск, открытие карточки товара, добавление в корзину и др.) инициируют отправку ивентов в трекер
В корзину
Немецкий дом
Перейти в магазин
Добавить в Избранное
Добавить в список
Похожие товары
Event processing
Tracker (ClickHouse)

2

Интерфейс конструктора сегментов


Действие 1
Выберите действие, которое должен совершить пользователь, чтобы попасть в сегмент или введите SQL запрос
Действие
Просмотр товара на PDP
Параметр *
Категория
Условие *
Один из выбра...
Значение параметра
РЫБА, МОРЕПРОДУКТЫ
ДОБАВИТЬ ПАРАМЕТР
ДОБАВИТЬ ДЕЙСТВИЕ
Исключить пользователя из сегментов
#1956 ИНТЕРЕСОВАЛИСЬ ИЛИ ПОКУПАЛИ ТАТУ
Тип сегмента
Динамический
Статический
Период сбора данных о пользователях
ПОСЛЕДНИЕ 24 ЧАСА
ПОСЛЕДНИЕ 3 ДНЯ
ПОСЛЕДНИЕ 7 ДНЕЙ
ПОСЛЕДНИЕ 14 ДНЕЙ
СОХРАНИТЬ

Создание сегмента на основе данных Ozon трекера

Условия Запросы

Источник - ClickHouse трекера
Период сбора данных о пользователях - Последние 3 дня
Событие - Просмотр товара на PDP
Категория - Рыба, морепродукты

Тип сборки сегмента

Сборка по userId

Пользователь исключается из сегментов

#1956 Интересовались или покупали ТАТУ

Глубина сегмента

РАССЧИТАТЬ

Название сегмента (минимум 10 символов)

Название *
Интересовались морепродуктами

Время существования пользователя в сегменте

Время в днях *
5

ИЗМЕНИТЬ СОЗДАТЬ СЕГМЕНТ

Создание сегмента на основе данных Ozon трекера

Условия попадания пользователя в сегмент

Условия | Запросы

Источник - ClickHouse трекера

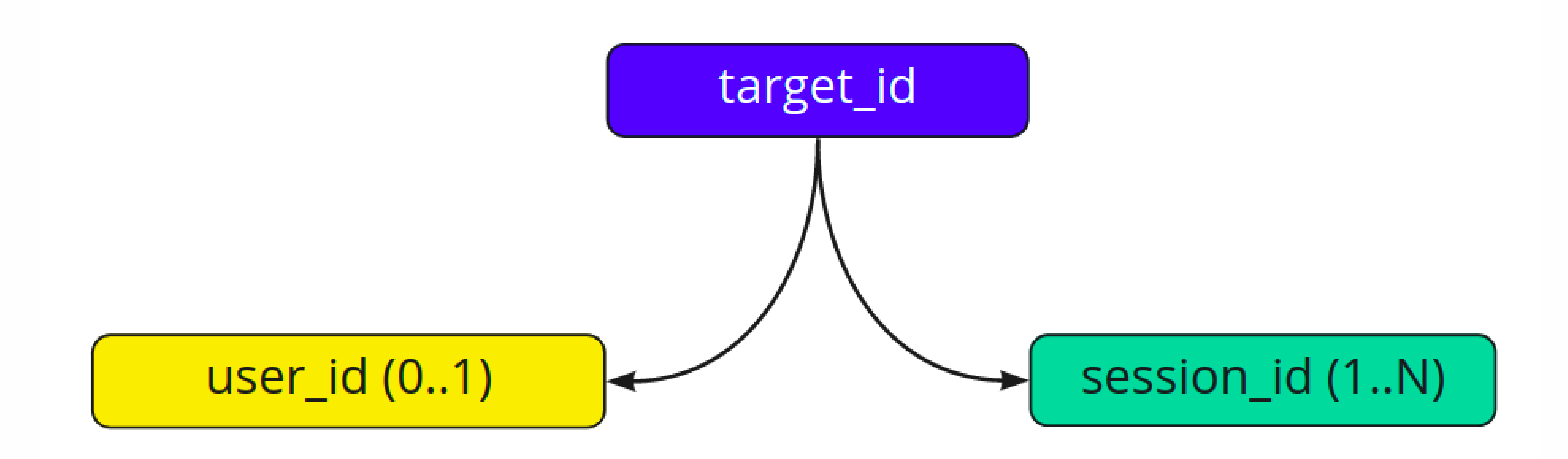
SQL запрос:

```
SELECT
  user_client_id,
  count() event_counter
FROM
  (
    SELECT
      user_client_id
    FROM
      events
    WHERE
      attributes_namespace = 'bx'
      AND (
        (
          action_type = 'view'
          AND object_type = 'product'
          AND action_widget = 'pdp-widget'
        )
        AND date BETWEEN '2021-04-26'
        AND '2021-04-29'
        AND hasAny(
          dictGetHierarchy(
            'navigation_category',
            toUInt64(
              dictGet(
                'sku',
                'navigation_category_id',
                toUInt64(object_sku)
              )
            )
          ),
```

ИЗМЕНИТЬ | СОЗДАТЬ СЕГМЕНТ

3

Архитектура



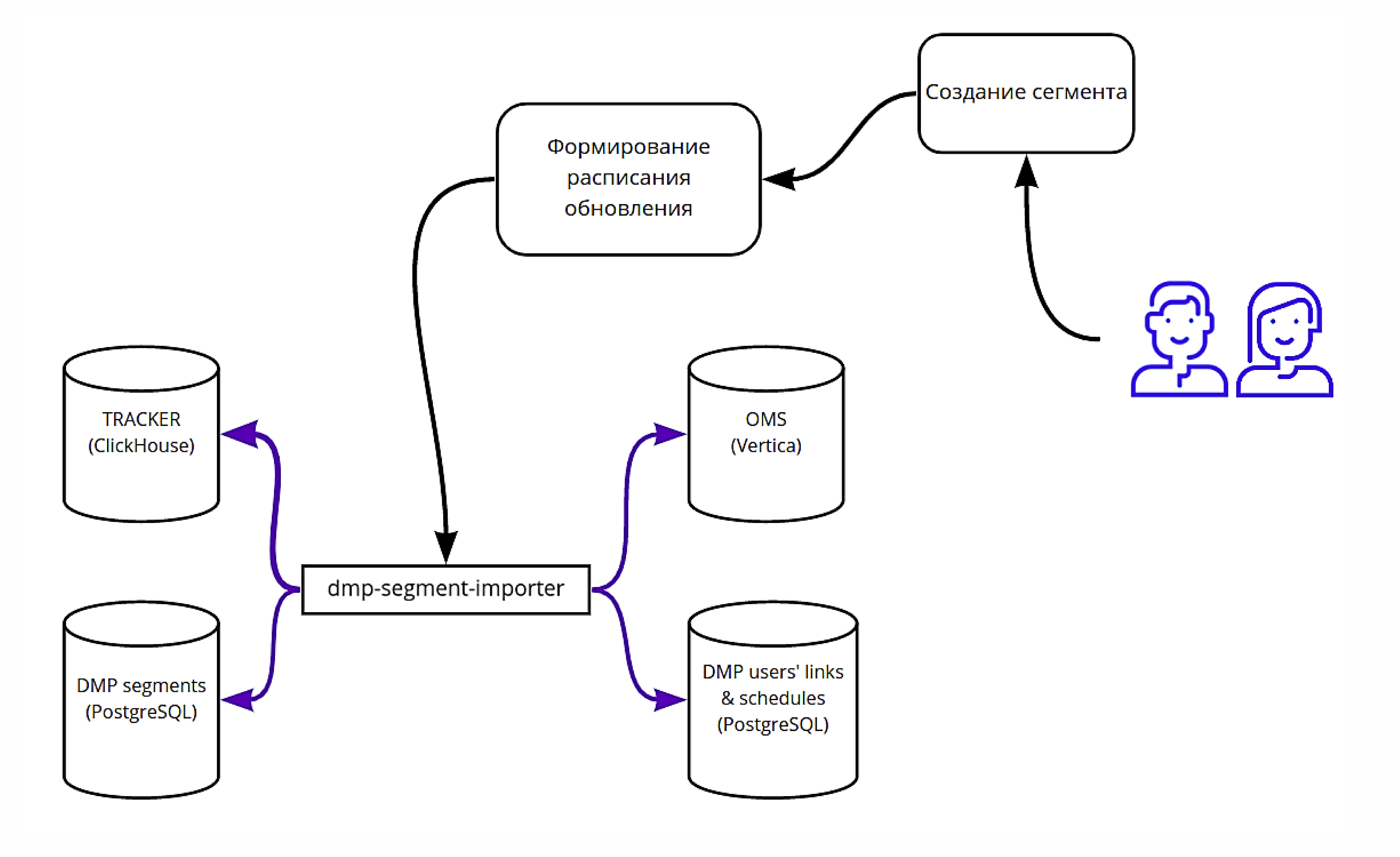
target_id связывает сессии и user_id между собой

Для DMP одна связка – один пользователь

Пример сегментов:

```
table segments
(
   dmp_id,
   segments jsonb
);
```

```
{
  "66": {  -- номер сегмента
    "source": "ozon tracker page view",
    "recency": 1618659304, -- время добавления в сегмент
    "expires_at": 1634470504, -- время экспирации = recency + TTL
  },
  "365": {
    "source": "ozon tracker page view",
    "recency": 1618659304,
    "expires_at": 1650195304,
  }
}
```


Создание сегмента
Формирование расписания обновления
TRACKER (ClickHouse)
OMS (Vertica)
dmp-segment-importer
DMP segments (PostgreSQL)
DMP users' links & schedules (PostgreSQL)

```
{
  "formula": "A|B",
  "requests": {
    "A": {
      "body": {
        "filter": {
          "$and": [
            {"$event_alias": "view_pdp_product"},
            {"$date_alias": "LAST_60_DAYS"},
            {"$field": "properties_brand_id","$value": 87314531,"$operator": "$eq"}
          ]
        }
      },
      "source": "tracker"
    },
    "B": {
      "body": {
        "filter": {
          "$and": [
            {"$event_alias": "favorite_product"},
            {"$date_alias": "LAST_60_DAYS"},
            {"$field": "properties_brand_id","$value": 87314531,"$operator": "$eq"}
          ]
        }
      },
      "source": "tracker"
    }
  },
  "selection_entity": "USER_TYPE_ID"
}
```

Пример запроса для ClickHouse


```
SELECT user_client_id,
       count() event_counter
FROM (
      SELECT user_client_id
      FROM events
      WHERE attributes_namespace = 'bx'
        AND (
              (action_type = 'view' AND object_type = 'product' AND action_widget = 'pdp-widget')
              AND date BETWEEN '2021-03-04' AND '2021-05-03'
              AND has([87314531], dictGetUInt64('sku', 'brand_id', toUInt64(object_sku)))
          )
       )
GROUP BY user_client_id
```

Запрос в ClickHouse, который состоит из подзапросов по каждому действию (событию), выполняется гораздо медленнее, чем параллельное выполнение каждого подзапроса.

Лучше отправить несколько мелких запросов и потом применить алгоритм сортировочной станции по формуле для полученных пользовательских id’ов.

Такой подход позволяет использовать разные источники данных для построения сегментов.

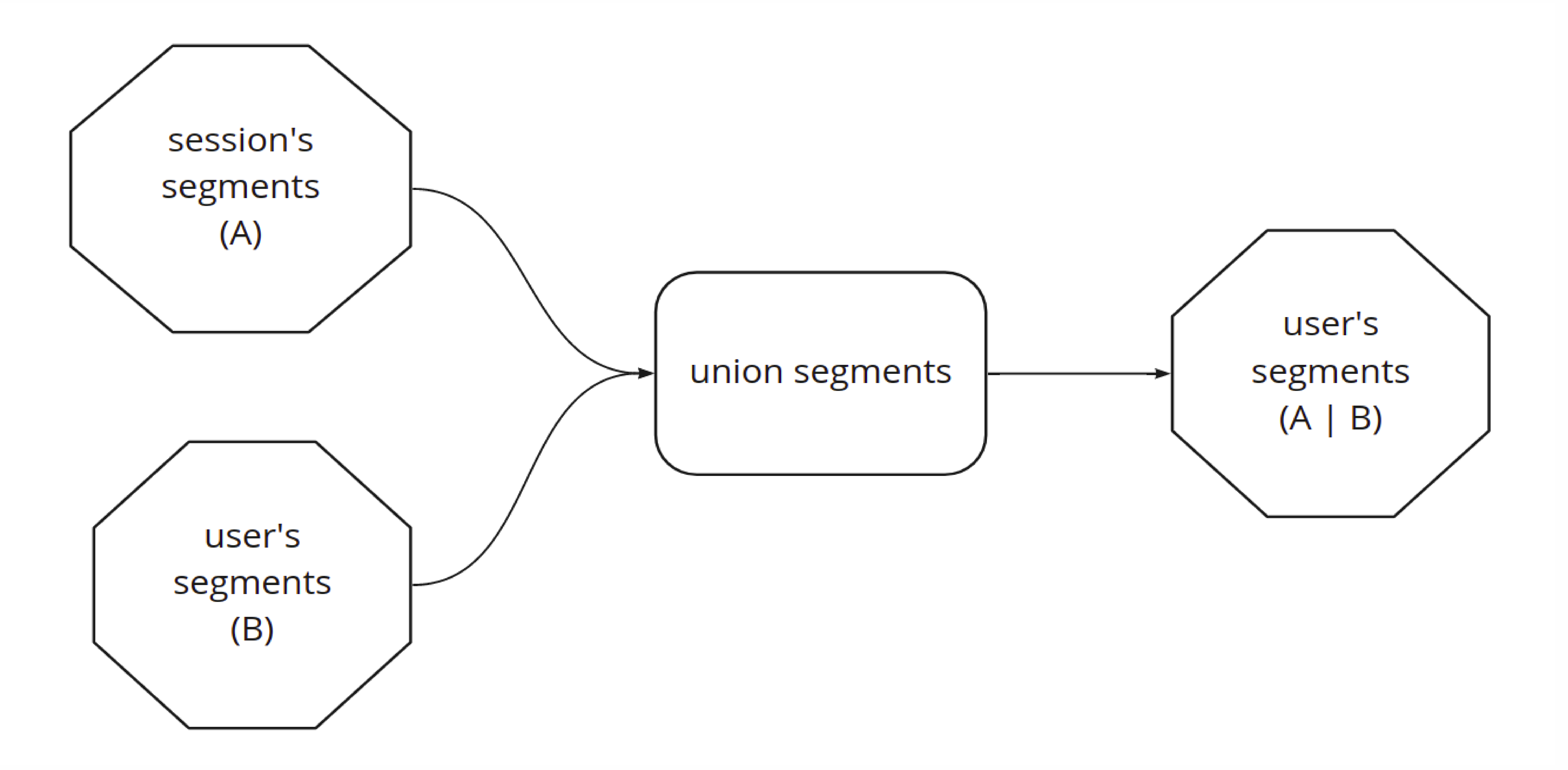
Сборка real-time-сегментов



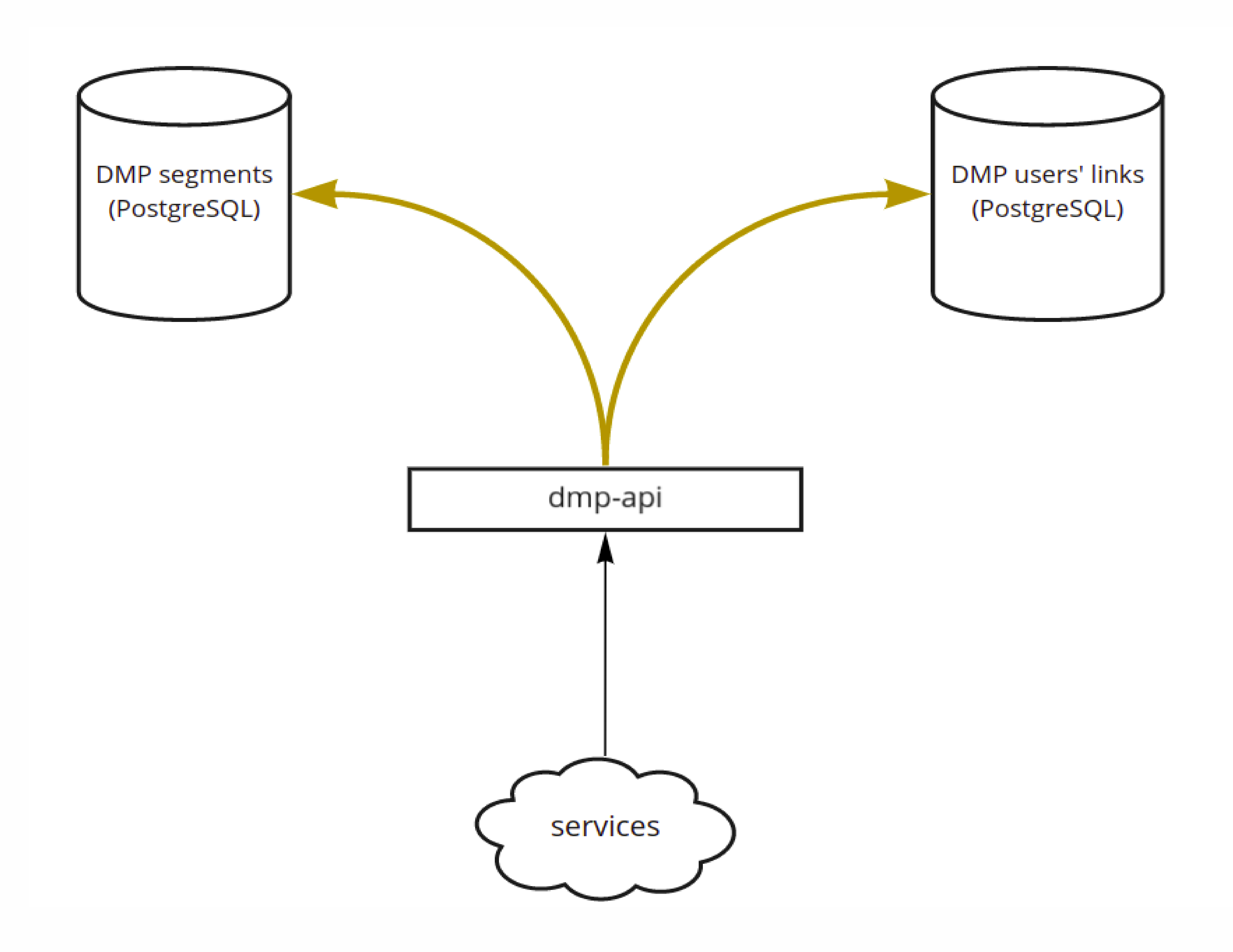
Объединение сегментов

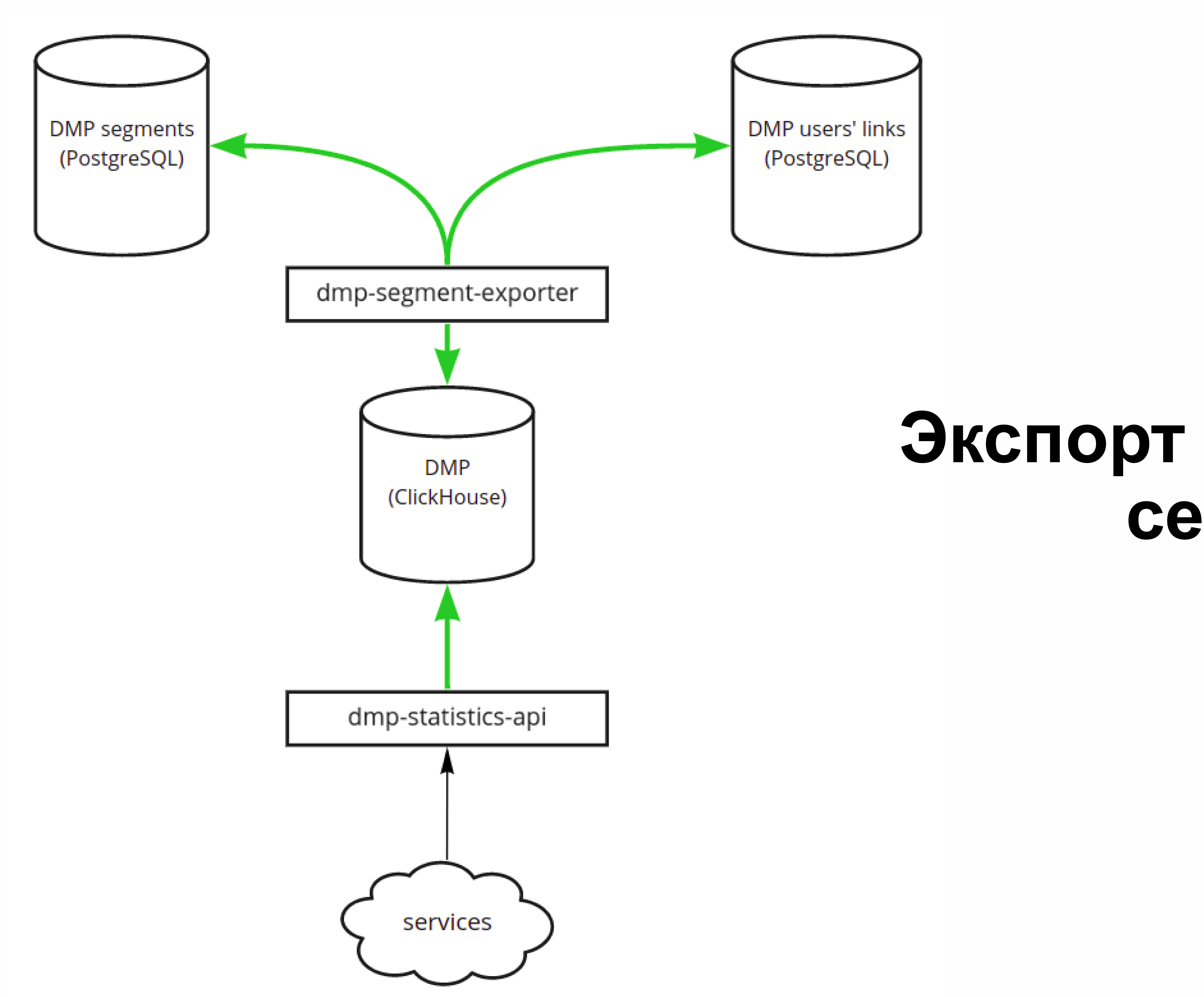
Выполняется при аутентификации пользователя

Сегменты сессии объединяются с сегментами пользователя



Выдача сегментов





Экспорт и пересечение сегментов

dmp-statistics-api

Оперирует сегментами как множествами.

Получение количества пользователей по формуле на определенную дату:

```
curl \
  -X POST "http://dmp-statistics-api.bx.stg.s.o3.ru:80/v1/segments/expression/quantity" \
  -d "{ \"formula\": \"372 & 536 & 576\", \"targetDate\": \"2020-06-15T08:03:08.485Z\"}"
```

Ответ:

```
{
  "quantity": 531789,
  "targetDate": "2020-06-15T00:00:00Z"
}
```

Примеры:

– "(372 & 536) | (600 & 602)" — объединение двух пересечений
– "372\536" — все, что есть в 372, но нет в 536

Получение списка пользователей в сегменте:

```
curl \
-X POST "http://dmp-statistics-api.bx.stg.s.o3.ru:80/v1/segment/users"
-d "{ \"limit\": 100, \"segmentId\": \"376\"}"
```

Запрос в DMP (ClickHouse) для получения пересечения трех сегментов: "1&2&3"

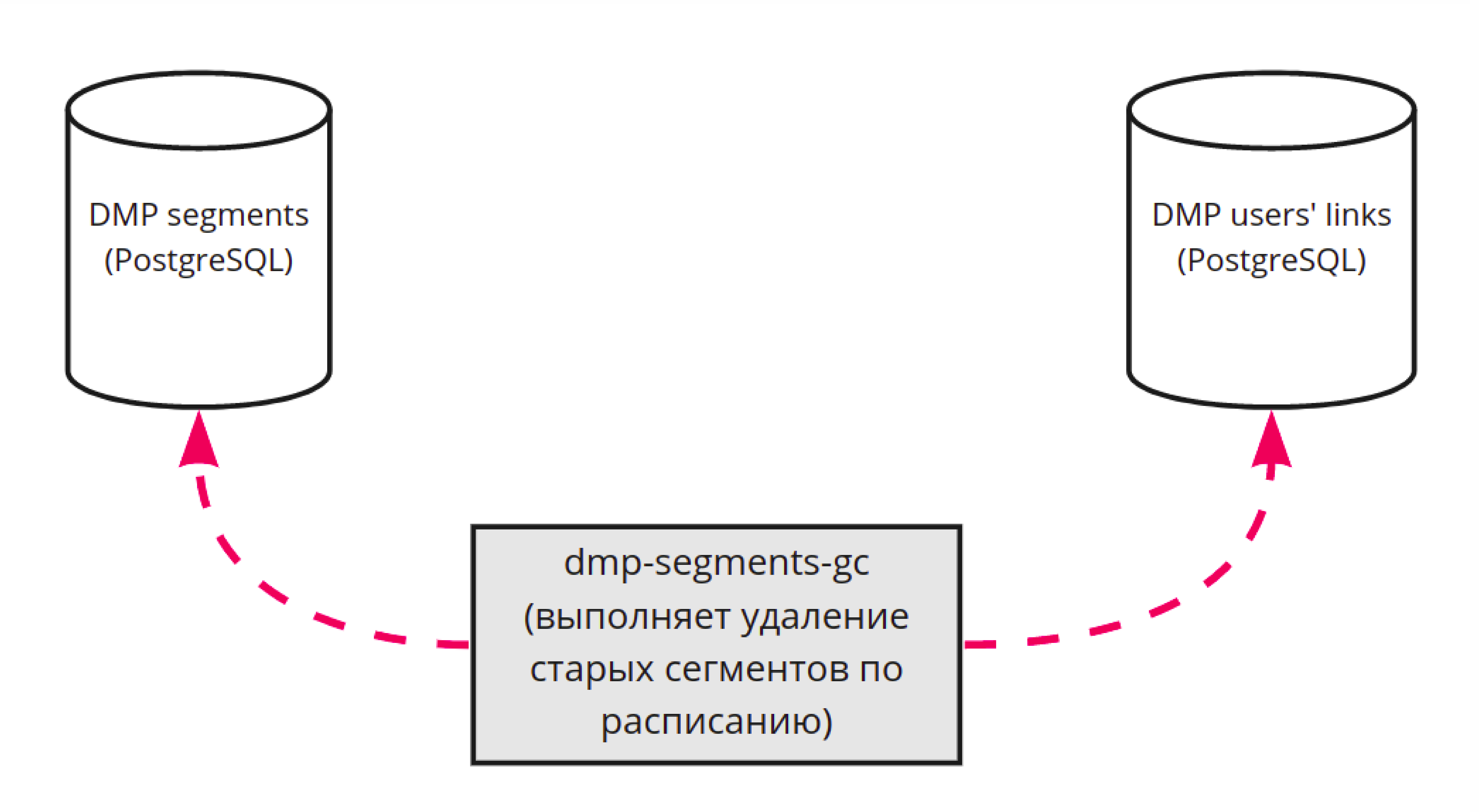
```
SELECT count()
FROM (SELECT dmp_id
      FROM dmp_segments_history
      WHERE segment_id = 1
        and date = 'date'
        AND dmp_id IN (SELECT dmp_id FROM dmp_segments_history

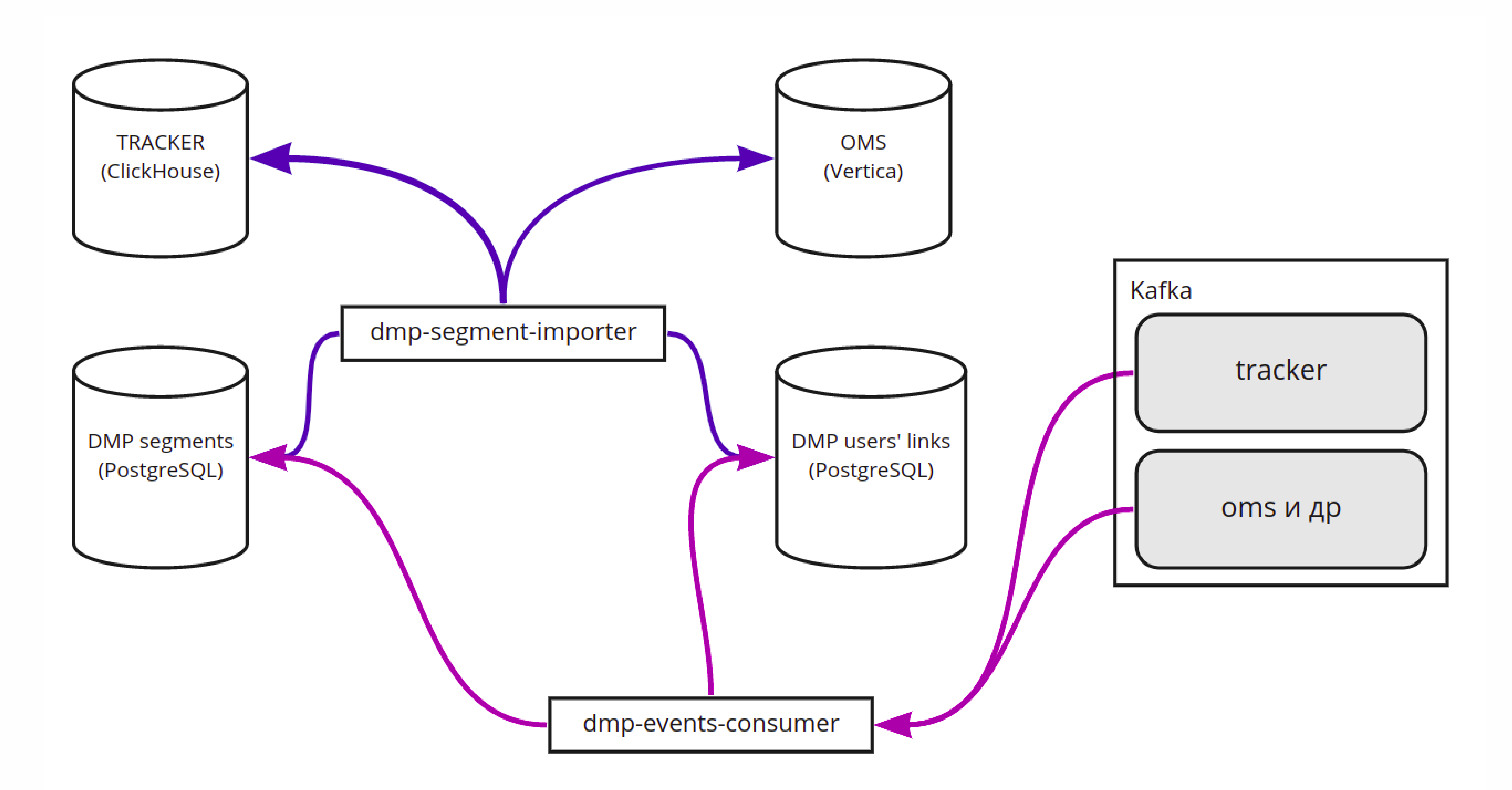
WHERE segment_id = 2 and date = 'date')
        AND dmp_id IN (SELECT dmp_id FROM dmp_segments_history

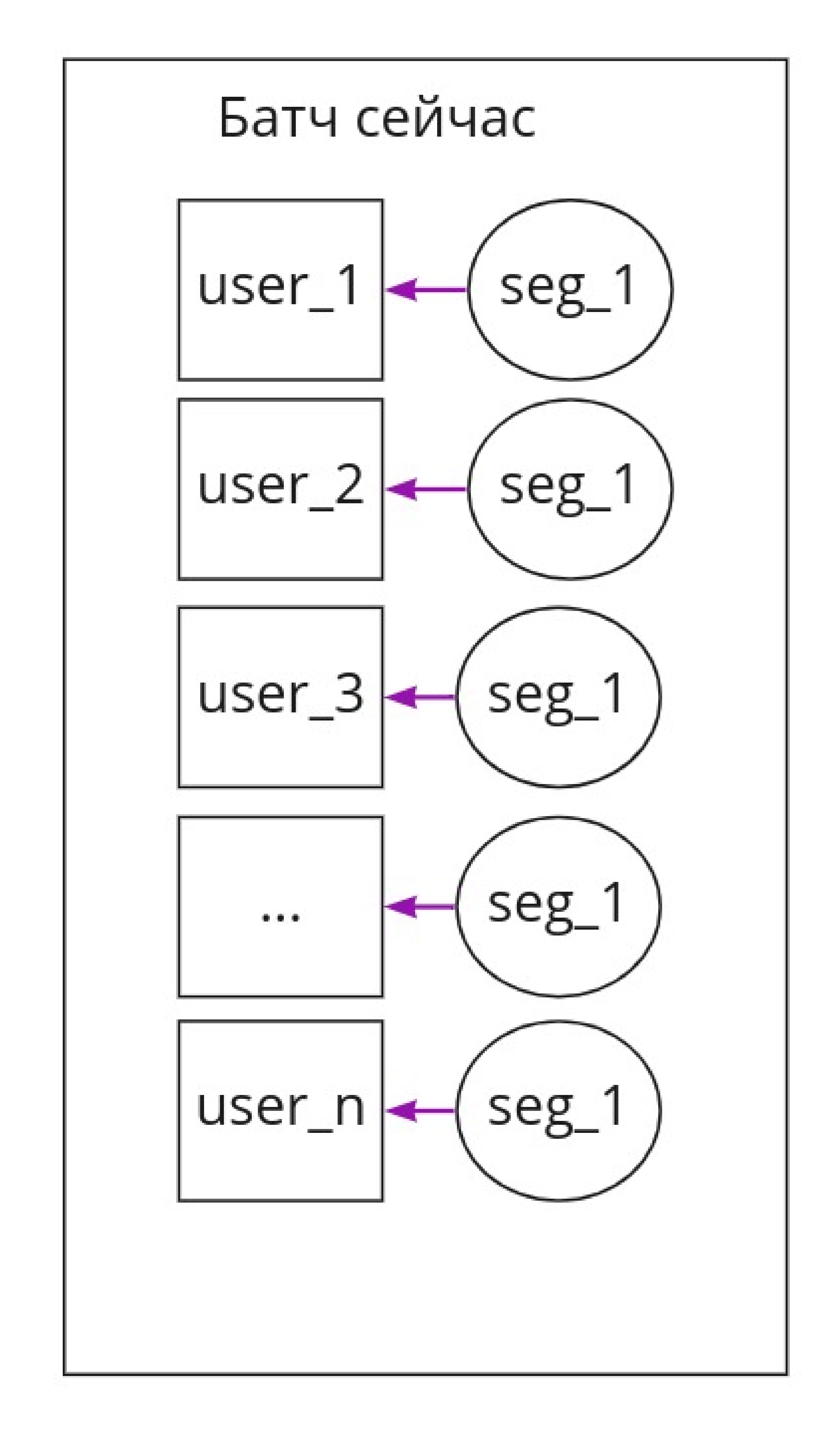
WHERE segment_id = 3 and date = 'date'))
```

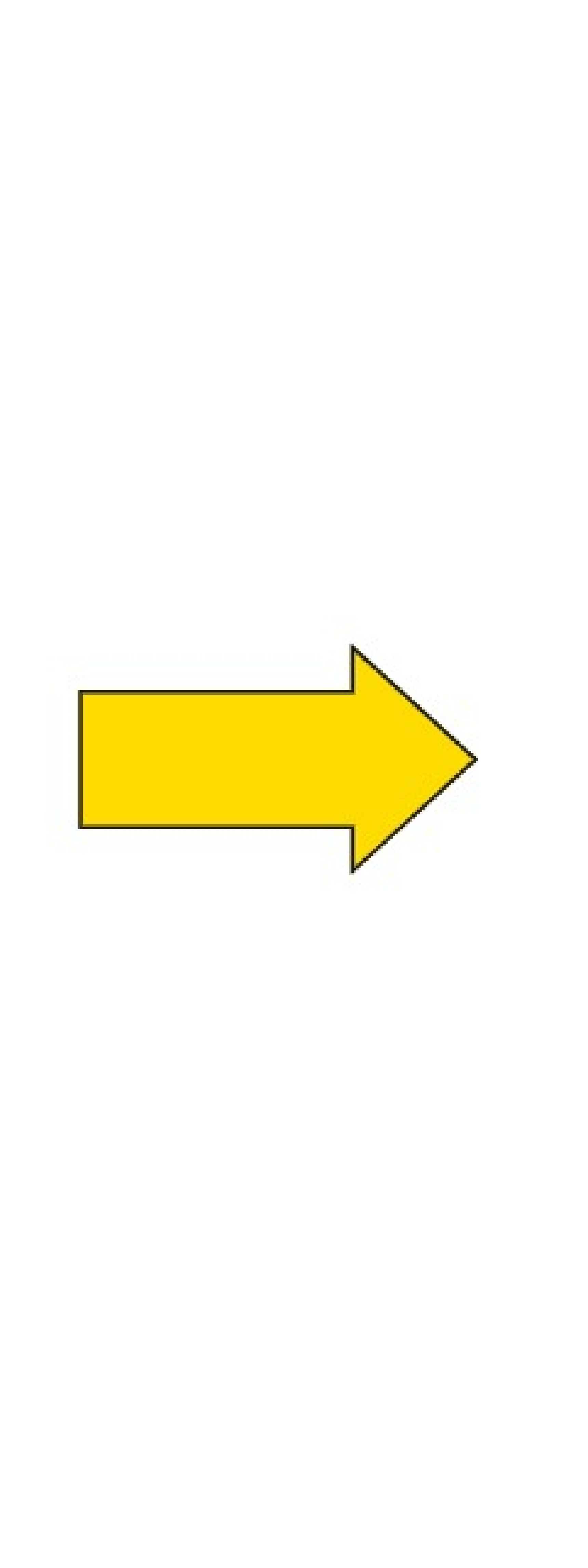
Большие JSON’ы стали проблемой #1. Тратится много процессорного времени на распаковку и сжатие JSONB.

Деградация скорости записи и чтения из-за роста jsonb.

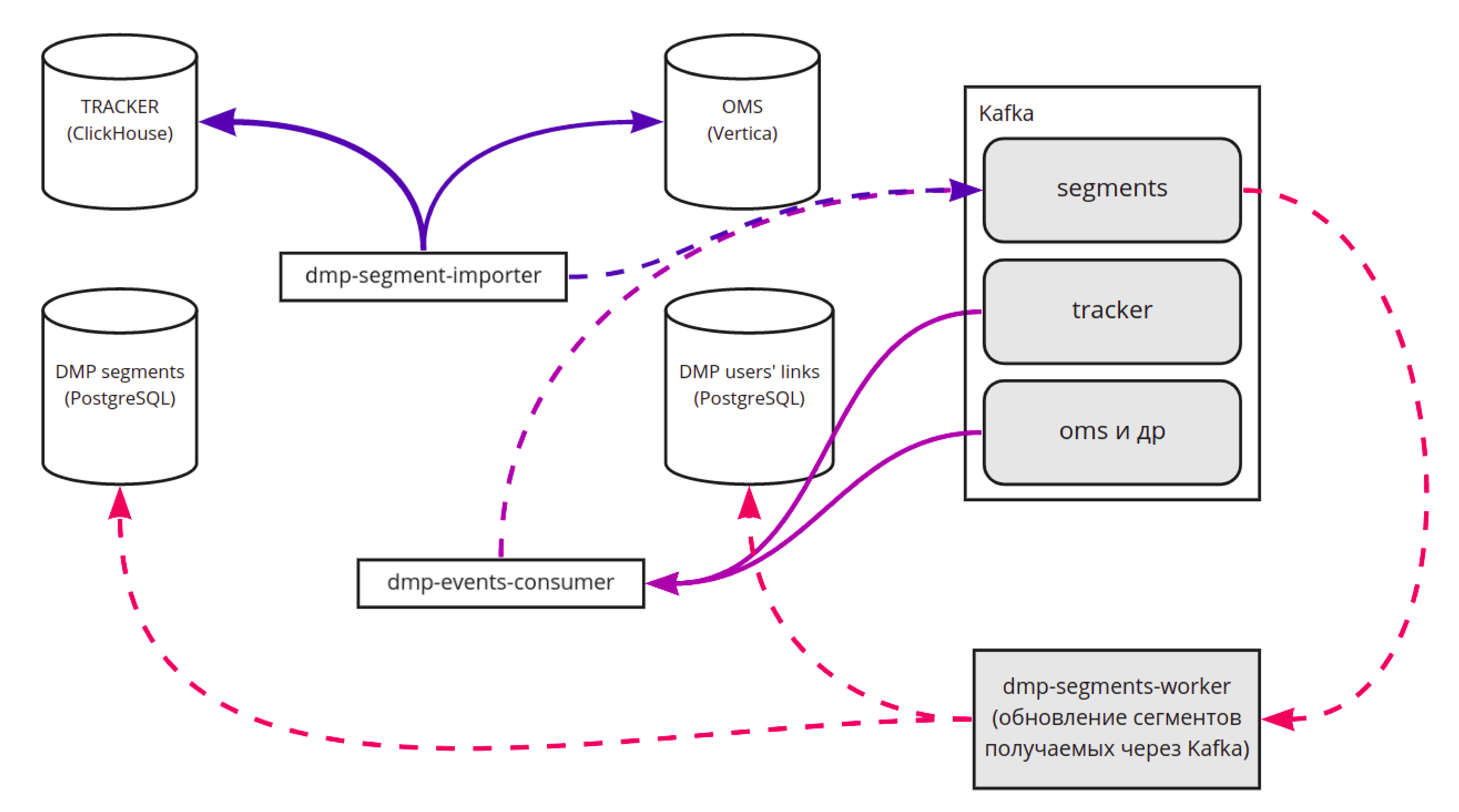

DMP segments
(PostgreSQL)
DMP users' links
(PostgreSQL)
dmp-segments-gc
(выполняет удаление
старых сегментов по
расписанию)

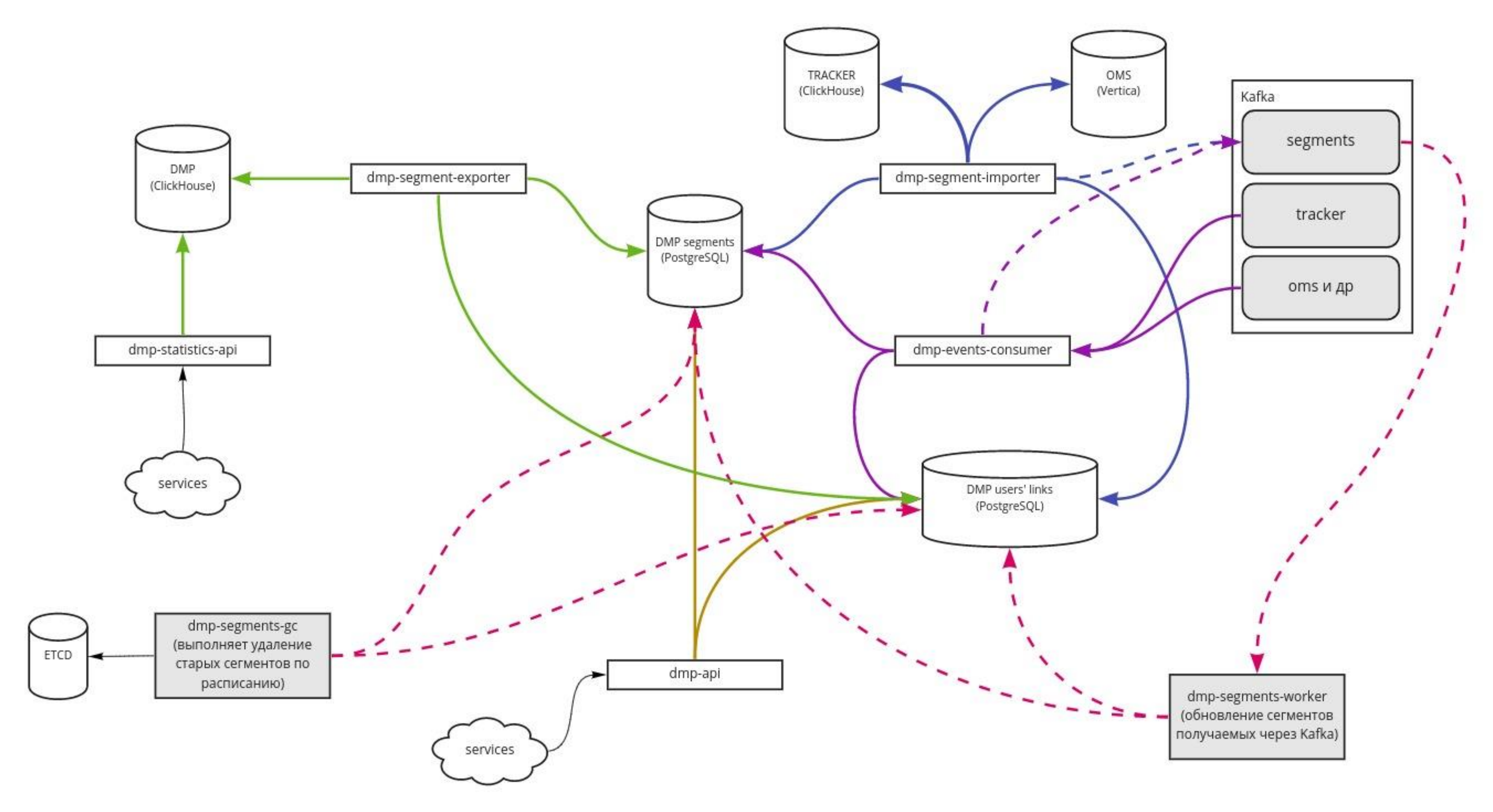

TRACKER
(ClickHouse)
OMS
(Vertica)
dmp-segment-importer
DMP segments
(PostgreSQL)
DMP users' links
(PostgreSQL)
Kafka
tracker
oms и др
dmp-events-consumer


Батч сейчас
user_1
seg_1
user_2
seg_1
user_3
seg_1
...
seg_1
user_n
seg_1



Батч dmp-segments-worker
user_1
seg_1
seg_2
seg_3
user_2
seg_1
seg_2
seg_3
user_3
seg_1
seg_2
seg_3
...
seg_1
seg_2
seg_3
user_n
seg_1
seg_2
seg_3


TRACKER
(ClickHouse)
OMS
(Vertica)
Kafka
segments
tracker
oms и др
dmp-segment-importer
DMP segments
(PostgreSQL)
DMP users' links
(PostgreSQL)
dmp-events-consumer
dmp-segments-worker
(обновление сегментов
получаемых через Kafka)


TRACKER
(ClickHouse)
OMS
(Vertica)
Kafka
segments
tracker
oms и др
DMP
(ClickHouse)
dmp-segment-exporter
DMP segments
(PostgreSQL)
dmp-segment-importer
dmp-statistics-api
dmp-events-consumer
services
DMP users' links
(PostgreSQL)
ETCD
dmp-segments-gc
(выполняет удаление
старых сегментов по
расписанию)
dmp-api
dmp-segments-worker
(обновление сегментов
получаемых через Kafka)
services

- **dmp-events-consumer** — построение real-time-сегментов по ивентам из kafka + merge-сегментов;
- **dmp-api** — отдает сегменты по запросу с user_id или session_id;
- **dmp-segments-importer** — конструктор сегментов + API по добавлению пользователей в сегменты;
- **dmp-segments-exporter** — экспорт сегментов в ClickHouse для формирования статистики;
- **dmp-statistics-api** — получение глубины сегментов и их пересечений за произвольную дату.

Оптимизация (WIP)

- **dmp-segments-gc** — удаление сегментов с истекшим TTL. Чистка нужна для уменьшения потребляемого места на диске и ускорения операций RW над jsonb, который содержит сегменты.

- **dmp-segments-worker** — группировка нескольких сегментов по пользователю в батче с целью уменьшить количество операций записи в PostgreSQL.

Проблема 1:

Частая смена сессий тестовых аккаунтов

Торможение мержа сегментов для больших связок в id_map

Решение:

Ограничили количество сессий для одного target_id + изменили алгоритм мержа.

Проблема 2:

С одним мастером уперлись в лимиты (17К) IOps. Таблица segments быстро росла → bloats. Стали делать pg_repack раз в месяц (4–6 часов и не всегда удачно с первого раза. Для pg_repack желательно снизить интенсивность записи.

Решение:

Пошардились на 12 пар (master-sync реплика).

Проблема 3:

Лишние перезаписывания сегментов.

Обновление сегмента чаще всего выполняется раз в сутки. Например, фильтрация может выполнятся за последние 90 дней.



Решение:

Стали выполнять пересечение отфильтрованного набора пользователей с набором ~ DAU -> сократили количество апдейтов по каждому сегменту.



До шардинга


IO / s
20 K
15 K
10 K
5 K
мастер
экспорт в ClickHouse
Load Average
35
30
25
20
15
10
5
0

После шардинга




Load Average
6
5
4
3
2
1
0
мастер
реплика


UPDATE (rows/second)
rows / second
3.50 K
3 K
2.50 K
2 K
1.50 K
1 K
500
0
2021-05-12 22:54:05
public.segments: 2.80 K

Выводы

- Сразу делать real-time-сборку сегментов. В этом случае надобность в пересборке практически отпадает;
- Чем больше JSONB, тем сильнее ощущается деградация скорости чтения/записи (CPU, disk IO). Встречаются экземпляры по 19Кбайт;
- Обновлять сегменты лучше батчами (3–4K пользователей);
- При построчной модели можно столкнуться с сильной деградацией чтения из-за фрагментированности данных. JSONB лучше, если сегментов много;
- Лучше несколько маленьких таблиц, чем одна большая -> более быстрое выполнение автовакуума, скорость накатывания дампа (проще обслуживать).

Спасибо за внимание!

Евгений Чмель

 echmel@ozon.ru



